COLLISION AVOIDANCE FOR ATEROS ROBOTIC SYSTEM

Adam Ligocki
Doctoral Degree Programme (2), FEEC BUT
E-mail: xligoc01@stud.feec.vutbr.cz

Supervised by: Luděk Žalud
E-mail: zalud@feec.vutbr.cz

Abstract: This paper describes the details of a collision avoidance algorithm for an ATEROS robotic system. The solution, developed and tested on the Orpheus robotic platform is based on a Velodyne HDL-32E laser scanner. The LiDAR point cloud input data are filtered to remove data redundancy and clustered to separate possible collision objects from the background. Based on prior environment knowledge and the current LiDAR scan, the surrounding occupancy grid map is estimated, and the planned path is validated against possible collision. In the case of a non-zero probability that the robot collides with an obstacle, a new path is proposed by the A* algorithm. Subsequently, the newly estimated waypoints are relaxed, and the mission plan is updated.

Keywords: Data Acquisition, Camera, LiDAR, IMU, GNSS, Odometry, Data Fusion

1 INTRODUCTION

This paper characterizes a new collision avoidance software solution for the Orpheus robot [6], a member of the ATEROS robotic system developed at the Department of Control and Instrumentation at the Faculty of Electrical Engineering and Communication, Brno University of Technology.

The ATEROS robotic structure [2] is primarily designed to operate in environments dangerous for human beings, such as radiation-contaminated or chemically polluted zones. The system is fully applicable within Urban Search and Rescue (USAR). The ATEROS concept comprises more types of robots to cover different mission requirements. Apart from Orpheus, which is an army-oriented product, several other robots are available, including Morpheus, a civil modification of Orpheus [5], Scorpion, designed to be capable of upstairs movement, Brontes, a lightweight and highly mobile robot and Uranus, a quad-copter with a high-precision terrain scanning technology.

2 SOLUTION CONCEPT

A typical Orpheus task is to explore an area contaminated by unknown chemical or radioactive substances. In most cases, the operator who controls the Orpheus setup has prior knowledge of the area to be investigated. The relevant maps or satellite images are used to plan the robot's path or other mission aspects. But even such instruments may not eliminate unexpected hampering factors, including for example, obstacles or impassable terrain. The collision avoidance mechanism is then used to overcome these problems.

The actual architecture of the device comprises three individual parts (Fig. 1). The first of these three is the Orpheus robotic structure, an encapsulated platform with multiple integrated systems to detect and manage radiation and chemical contamination. The second part embodies a waypoint server deployed on a separate computer. This server provides the Orpheus with information about the planned path and mission progress. The Orpheus and the server are connected via an IP protocol based on wireless radio connection. The third part of the architecture constitutes the core of the

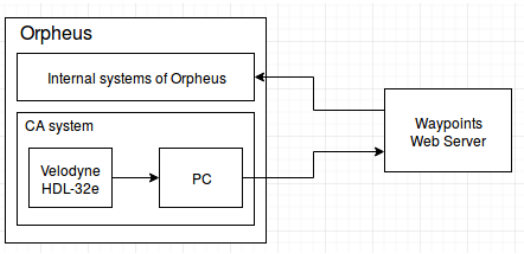

Figure 1: Schematic of basic system concept.

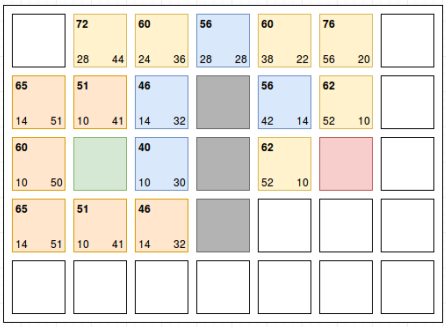

Figure 2: Example of A* path searching. Green - start point, Red - end point, Yellow - searched cells, Orange - frontiers, Blue - estimated path. Each searched cell has estimated cost from start point (left bottom) and distance from end point (right bottom). Final cost (left top) is sum of both previous distances.

research presented herein. It consists of a PC with a Velodyne HDL-32E LiDAR sensor and a precise differential GNSS receiver. This configuration continuously scans the surrounding area, parallelly it is communicating with the waypoint server to obtain information of the latest progress of the mission. If the LiDAR sensor detects an unexpected obstacle in the robot's path, the collision avoidance algorithm recalculates the pathway and transmits the result to the waypoint server. By periodical updating, the new waypoints will be incorporated into the Orpheus mission plan.

3 PATH SEARCHING

The principle of the collision avoidance problem rests in searching a path from the starting point A to the endpoint B. For this purpose, a graph-based solution is usually used in the given context. The set of basic graph-based searching algorithms includes for instance Breadth First Search, Dijkstra's algorithm, and the A* algorithm. In the present section, the A* algorithm, that has been used for my purpose, is briefly described.

3.1 A* ALGORITHM

The A* algorithm [1] contributes additional functional improvement compared to Dijkstra, giving each cell the heuristic knowledge of how far this cell is from the endpoint. Thus, now that the "frontier " area has expanded, the final distance number of each cell comprises the distance from the start point in Dijkstra's design plus the heuristic distance to the end point. This then helps the algorithm to invariably attempt to search for a path in the direction of the end point.

An example of the A* algorithm is shown in Fig. 2

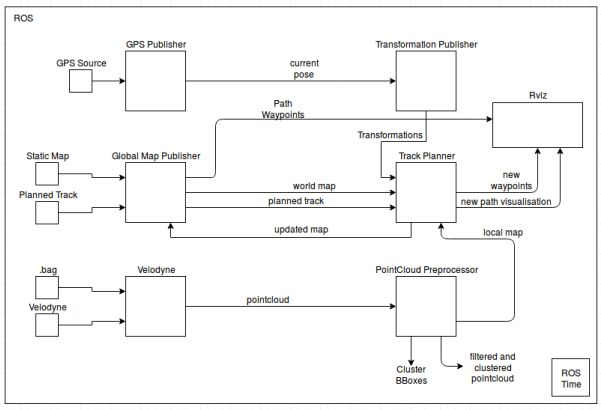

Figure 3: Collision avoidance data flow.

4 CORE SOLUTION

4.1 DATA INPUTS

At the input of the entire collision avoidance system there is a differential GNSS receiver and a Velodyne HDL-32E LiDAR scanner.

As is shown in Fig. 3, the data from the scanner pass through the Velodyne node, which transforms the RAW data into a point cloud message at the frequency of 10Hz. This message is passed to a Point Cloud Preprocessor (PCP). This node ensures the basic point cloud filtering and clusterization. At the output of the PCP there is message that contains the position and dimensions of detected obstacles.

The point cloud processing is performed by the Point Cloud Library (PCL) [3] framework, an open source tool with a wide range of functionalities in the field of point cloud processing, like filtering, clusterization, and registration.

The GNSS data are received as UDP socket communication between a GPS receiver as the server and the GPS Publisher (GP) as a client. The data are sent in the form of a NMEA standardized message. The incoming strings are parsed, and the information about the absolute global position and robot orientation is extracted and proceed into a Transformation Publisher (TPu) node.

The employed TPu node aggregates the GNSS position and orientation data, and it calculates the transformation (1) between the preconfigured global reference point and the current robot's position as well as the static transformation between the robot's origin and the Velodyne's position. All of the transformations between frames are published in the back-end.

$$P_1 = H_{12} * P_2 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} \tag{1}$$

The third data source in the system is the Global Map Publisher (GMP) node, which loads the static, preprepared map data of the mission area from a local storage and handles the planned path, downloaded through a UDP socket from the waypoint server.

The information proceeds to the Track Planner (TPl) node, the core of the entire system, where all the planning algorithm are executed.

4.2 MAP FUSION

The Track Planner node collects all preprocessed data, such as the filtered point cloud, transformation tree, and global map. It gathers all information about the environment and attempts to fuse the items into a single environment model where the current path is verified. In cases of possible collision the new way is proposed and propagated into the waypoint server.

The Track Planner node utilizes two types of maps. The first one is the global map, referenced to the world origin. The map is continuously received from the GMP node and stored in a planar occupation grid format with the cell size of 0.5m. The second type of maps is the local one referenced to the robot's origin. Such maps are also of the grid occupation structure, but for every incoming filtered point cloud message, a new local map is created by projecting all points of the point cloud onto the ground. The cells where a point has been projected are occupied, with the rest of them left free. In this way the constructed local map is fused with the global one. All of the occupied cells from the global map are transformed from the world's to the robot's coordinates, and the local map is enriched with prior global knowledge. The new local map is later projected into the global one, and the new global model is propagated into the GMP node. Here, the loop closes. The memory is also realized in the described manner to enable the introduction of newly detected obstacles into the global map model.

Further, the mechanism of forgetting is implemented. All obstacles written into the static global map during the startup process are permanent. Conversely, obstacles detected by the Velodyne scanner are temporal and if an obstacle disappears, the probability of a collision object presence in the occupancy grid continuously decreases down to zero so that the path could be planned through the cell again.

4.3 PATH PLANNING

Every time a new local fused map is created, the current planned path is validated by checking every waypoint inside the local map borders and paths between them. Each section of the path is projected into the current local map, and if an obstacle is detected along this section, the path planning algorithm proposes a new way. If any two waypoints defining a path section are projected into a cell where an obstacle has non-zero probability, the path-finding algorithm will continue to test other waypoints and their collisions with obstacles until two valid waypoints are found. At this stage, a new path section is proposed, and the original sector between the two waypoints is replaced with a newly defined path.

The next process within path planning is path relaxation. Each newly estimated path is post processed to remove sharp shapes. This waypoint relaxation is performed using the iterative minimizing path cost function according to equation 2 by [4],

$$\begin{aligned} (x_i - y_i)^2 &\rightarrow min \\ (y_i - y_{i+1})^2 &\rightarrow min \end{aligned} \tag{2}$$

,which express that the distance x_i is the original waypoint position, the y_i is the newly estimated position and y_{i+1} is the next upcoming optimized waypoint. In this way we are trying to minimize the distance between neighbour waypoints (smooth the path) in the same time as we want to keep distance between original waypoint's position and the new one (keep the shape of the path).

In the next phase, the new waypoints are propagated into the waypoint web server, and the mission plan is updated.

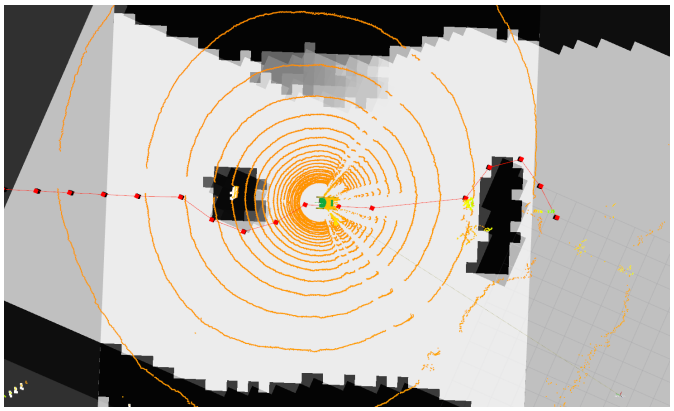

Figure 4: New path after collision detection and new path estimation.

5 CONCLUSION

This paper discusses the details of a collision avoidance developed for ATEROS robotic system. The entire solution utilizes a Velodyne HDL-32E LiDAR scanner to detect possible collision objects. Based on scans, local map is created and fused with previously formed global one. After the map building process, where a map is represented as an occupancy grid, the current planned path kept by a remote server is validated and in the cases that the Orpheus robot might collide with a detected object, a new path is proposed by the A* algorithm. And propagated into mission plan server.

This functionality is integrated on a single platform with multiple other features, [5] embodies a unique rescue system usable in both the civil domain and the army sector.

ACKNOWLEDGEMENT

The completion of this paper was made possible by the grant No. FEKT-S-17-4234 - „Industry 4.0 in automation and cybernetics" financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[2] T. Lazna, T. Jilek, P. Gabrlik, and L. Zalud. Multi-robotic area exploration for environmental protection. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 240–254. Springer, 2017.

[3] R. B. Rusu and S. Cousins. Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[4] S. Thrun, W. Burgard, and D. Fox. Probabilistic robotics (intelligent robotics and autonomous agents series). *Intelligent robotics and autonomous agents, The MIT Press (August 2005)*, 2006.

[5] L. Zalud. Orpheus–reconniaissance teleoperated robotic system. In *16th IFAC world congress*, pages 1–6. Prague, Czech Republic Prague, Czech Republic, 2005.

[6] L. Zalud. Argos-system for heterogeneous mobile robot teleoperation. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 211–216. IEEE, 2006.